Basic Research in Computer Science

Some Complexity Problems on Single Input Double Output Controllers

Katalin M. Hangos
Zsolt Tuza
Anders Yeo

BRICS Report Series **RS-01-18**

ISSN 0909-0878 **2001**

**See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:**

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/01/18/`

Some Complexity Problems on Single Input Double Output Controllers

K. M. Hangos Zs. Tuza

Computer and Automation Research Institute, Hungarian Academy of Sciences, H–1111 Budapest, Kende u. 13–17, Hungary

and

Department of Computer Science, University of Veszprém

H–8200 Veszprém, Egyetem u. 10, Hungary

A. Yeo

BRICS*, Department of Computer Science, University of Århus

Ny Munkegade, bldg. 540, DK–8000 Århus C, Denmark

Version of 2001–5–18

Abstract

We investigate the time complexity of constructing single input double output state feedback controller structures, given the directed structure graph G of a system. Such a controller structure defines a restricted type of P_3-partition of the graph G. A necessary condition $(*)$ has been found and two classes of graphs have been identified where the search problem of finding a feasible P_3-partition is polynomially solvable and, in addition, $(*)$ is not only necessary but also sufficient for the existence of a P_3-partition. It is shown further that the decision problem on two particular graph classes — defined in terms of forbidden subgraphs — remains NP-complete, but is polynomially solvable on the intersection of those two classes. Moreover, for every natural number m, a stabilizing structure with Single Input m-Output controllers can be found in polynomial time for the system in question, if it admits one.

*Basic Research in Computer Science, Centre of the Danish National Research Foundation.

1 Introduction

The problems investigated in this paper originate from a well known and difficult engineering problem on designing a distributed control system of a complex plant. The source of the complexity problems lies in the fact that we aim at designing an optimal distributed control system structure, that is an optimal partitioning of the system variables.

Graph-theoretic description of process and control system structures is proposed in the earlier papers [3], [4]. With these definitions, the algorithmic problem statements of finding a distributed, stabilizing or disturbance rejective control structure — based on a given process structure — are introduced, both for the weighted and unweighted cases. For *single input single output* stabilizing and disturbance rejective controllers we proved that the existence of a control structure can be tested in polynomial time via constructing a matching that covers the set of state variables. Moreover, in this case the formulated optimal (weighted) controller structure selection problem has also been shown to be solvable in polynomial time.

We have also shown that already the *single input triple output* control structure selection problems are computationally hard for both of the stabilizing and disturbance rejective cases [3], [5].

The main subject of this paper is to investigate the problem of the *single input double output* control structure selection case. The problem will be handled in the form of restricted P_3-partitions of graphs. In addition, some results on single input *multiple output* controllers are presented, too.

The paper is organized as follows. The engineering problem and the corresponding combinatorial / algorithmic problems are introduced in Sections 2 and 3, respectively. A necessary condition for the existence of a solution is presented in Section 3.1 (completed with some more general ones in Section 3.2), and its sufficiency is proved in Section 4 for two particular classes of instances. The condition is not sufficient in general, however, as demonstrated by two simple examples called E and E'. Algorithmically, the corresponding graph partition problem is *NP*-complete, and it remains so even if we exclude all instances containing E — or, alternatively, E' — as a subgraph (Section 6). On the other hand, if both subgraphs E and E' are excluded, then the restricted graph class admits polynomial-time decision and search algorithms (Section 5). The last two sections deal with bipartite graphs. Some of the problems still remain algorithmically intractable (Section 7), but notably the existence

problem for Single Input m-Output controllers admits a polynomial-time solution (both for decision and search), as proved in Section 8.

2 Engineering problem statement

The engineering problem statement below explains the specialities of our combinatorial problem. The dynamics of a concentrated parameter nonlinear dynamic system (with or without controllers) can be described using the following state equation [7]

$$\frac{dy}{dt} = f(y, x)\,, \qquad y(0) = y_0 \tag{S}$$

$$\dim x(t) = r\,, \qquad \dim y(t) = p$$

where $y(t)$ is the state vector and $x(t)$ is the manipulable input vector at any time t. Note that both x and y depend on time, moreover x is assumed to be manipulable and acts as a cause for the state variations.

The variable structure of the above set of equations is described by a directed graph $G = (V, E)$, with a vertex partition $V = X \cup Y$ into two classes, as follows. The vertex set V consists of vertices associated to each of the state and input variables. There is a directed edge $y_j y_i \in E$ or $x_j y_i \in E$ present if the variable y_j or x_j appears in the argument of the right-hand side function f_i for $\frac{dy_i}{dt}$. It is important to note that the in-degree of a vertex x_j corresponding to an input variable is always equal to zero.

In order to modify the system behavior to satisfy a prescribed aim, say to stabilize the system, static state feedback controllers are most often applied. This is done by computing the value of the input variables $x_j,\ j = 1, \ldots, r$ using a given — possibly nonlinear — function g_j of some (or all) of the state variables

$$x_j = g_j(y_{j_1}, \ldots, y_{j_{p_j}}) \tag{C}$$

$$p_j \le p\,, \qquad j = 1, \ldots, r$$

characterized by the index set

$$I_j = \{\, j_1, \ldots, j_{p_j} \,\}\,, \qquad |I_j| = p_j\,.$$

The controllers (C) use p_j different state variables to compute the feedback: such a feedback controller is called *single input p_j output controller*.

3 Notions and problem statements

In this section we introduce some algorithmic problems, whose time complexity will be studied later on. We formulate them as *decision* problems; on the other hand, the corresponding *search* problems are of even greater practical importance. Therefore, for the instances proved to be polynomially decidable, we shall also desing polynomial-time algorithms that find feasible solutions if they exist.

The original engineering problem is equivalent to one on directed graphs. Nevertheless, we formulate two variants, one for digraphs and one for undirected graphs. As a matter of fact, in most cases, the "undirected version" can be reduced to the directed one by replacing each undirected edge y_iy_j with two oppositely oriented arcs y_iy_j and y_jy_i, and orienting each edge x_iy_j from x_i to y_j. (Then, in this "double-orientation," all undirected paths P_3 give rise to directed paths of length two, i.e. none of them gets lost. Therefore, a method finding a solution in the digraph obtained, yields a solution for the original undirected graph, too, in a natural way.) This convention will be applied in parts of the paper where we treat graphs and digraphs together.

We consider (di)graphs $G = (V, E)$, where V is the vertex set and E is the edge set, with the following structural assumptions:

- There is a given vertex partition $X \cup Y = V$, $X \cap Y = \emptyset$.
- The set X is independent in G.
- The "size condition" $|Y| = 2\,|X|$ holds.
- In the directed case, all X–Y edges are oriented from X to Y (but inside Y two vertices may be adjacent in both directions simultaneously).

We wish to decide whether there exists a vertex partition of G into $|X|$ disjoint paths of length 2, each of them containing precisely one vertex of X. Such paths will be called *feasible*, and it will be assumed throughout without loss of generality that *each vertex is contained in at least one feasible path* in G. The feasible paths need not be *induced* subgraphs, i.e. they may as well induce triangles in the graph (with possibly two oppositely oriented arcs inside Y). In digraphs, however, there is some restriction: if a path has its starting vertex in X, then its two edges have to be oriented consecutively, i.e. it must be a *directed* path. (A path with

its middle vertex in X has, of course, an alternating orientation.) If such a partition exists, we say that G is *P_3-partitionable*.

In this way, one can formalize an algorithmic problem, too, in the standard way. For short, we shall refer to this problem as 1X-2Y.

P_3-Partitioning for Single Input Double Output Controllers (1X-2Y):

Instance: A graph or digraph $G = (V, E)$, with vertex bipartition $X \cup Y = V$, where X is an independent set and $|Y| = 2\,|X|$.

Question: Is G P_3-partitionable?

The corresponding *search* problem takes the same instances, and asks for a feasible P_3-partition of G as solution.

For later use, we introduce here two further algorithmic problems arising in this context. They concern multiple-output controllers; in both definitions, m denotes any natural number (greater than 1; the case of $m = 1$ has been thoroughly studied in [3] and [5]).

Stabilizing Single Input m-Output Controller Existence (SSmE):

Instance: A graph or digraph $G = (V, E)$, with vertex bipartition $X \cup Y = V$, $|Y| \leq m\,|X|$.

Question: Does G have a vertex partition $V = V_0 \cup V_1 \cup \cdots \cup V_k$ (for some non-specified k), such that $V_0 \subset X$ and, for all $1 \leq i \leq k$, $|V_i \cap X| = 1$, $|V_i \cap Y| \leq m$, and there exists a directed path inside V_i from the vertex of $V_i \cap X$ to each $y \in V_i \cap Y$?

Stabilizing Single Input m-Output Controller Optimization (SSmO):

Instance: A graph or digraph $G = (V, E)$, with vertex bipartition $X \cup Y = V$, $|Y| \leq m\,|X|$.

Question: In the vertex partitions feasible in the sense of the SSmE problem, what is the smallest possible total number of edges joining the pairs V_i, V_j, counted for all $1 \leq i < j \leq k$?

As 1X-2Y is just a particular case of SSmE for $m = 2$ and $|Y| = m\,|X|$, and SSmO already assumes an answer to SSmE as well, all the three problems above turn out to be *NP*-complete by the results of Section 6

below. In fact, the optimization problem SSmO remains *NP*-complete when restricted to bipartite graphs, for all $m \geq 3$, as it has been observed in [3, 5]. On the other hand, we shall prove in Section 8 that the existence problem SSmE on bipartite graphs admits a polynomial-time solution for every m.

The problems SSmE and SSmO model the situation that the system may be stabilized by just a subset of the input variables. In addition, SSmO takes into account that if an x_i has been chosen to stabilize some state y_i while it is also adjacent to some y_j stabilized by another input variable, then x_i acts on y_j as a disturbance. (Similarly, disturbances may act along edges joining two states, too, which are stabilized by distinct controllers.) The elements of $V_0 \subset X$, however, which do not occur in the edges covering Y, need not be counted as disturbances.

Tuning the model further, in a linear system (or, in a system linearized around a steady state) the disturbances can be represented by *weighted* edges, and the overall goal would be to find a feasible distributed stabilizing structure in which the *total weight* of disturbances is *minimized*. We do not consider this weighted version here, however, because already the unweighted one is *NP*-complete.

3.1 The Neighborhood-Matching Condition

A necessary condition — which is not sufficient in general — for P_3-partitionability can be obtained as follows. For any $A \subseteq V$, let us denote by $N(A)$ the set of vertices in $V \setminus A$ joined to at least one element of A, and let $n(A) := |N(A)|$. Consider now a subset $A \subseteq X$. (Then $N(A) \subseteq Y$, since X is independent.) Let $m(A)$ denote the largest number of mutually vertex-disjoint edges starting in $N(A)$ and having the other endpoint in $Y \setminus N(A)$.

Proposition 1 *If G admits a P_3-partition, then*

$$n(A) + m(A) \geq 2\,|A| \qquad \forall\ A \subseteq X \tag{$*$}$$

Proof. The $|A|$ paths covering the vertices of A have precisely $2\,|A|$ vertices in Y, at most $m(A)$ of which can belong to $Y \setminus N(X)$, and all the others must be located inside $N(A)$. •

We shall call $(*)$ the *Neighborhood-Matching Condition*.

As observed by Kotlov [8], the following graph on six vertices shows that the condition $(*)$ alone is not sufficient for P_3-partitionability.

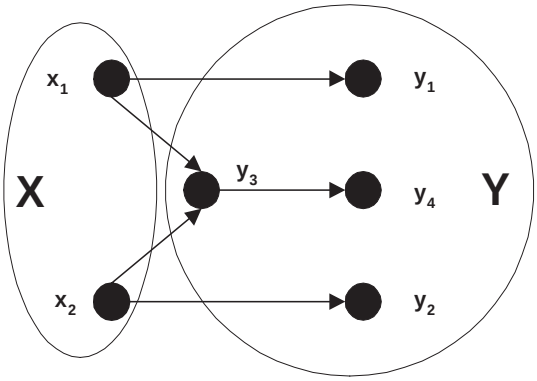

Figure 1: The graph E

Example 1 Set $X = \{x_1, x_2\}$ and $Y = \{y_1, y_2, y_3, y_4\}$, and let the edge set consist of the five edges x_1y_1, x_2y_2, and x_1y_3, x_2y_3, y_3y_4. We denote this graph by E as it is seen in Fig. 1.

Obviously, $(*)$ is satisfied in E. Moreover, it is easy to see that E does not admit any P_3-partition. For instance, one can observe that y_1 and y_2 are contained in unique paths of length 2, which share the vertex y_3. •

Historically, the first example of a non-P_3-partitionable graph satisfying $(*)$ — found by Holzman [6] — had nine vertices, but actually it turned out to be not minimal, in the sense that it contains E as an induced subgraph. On the other hand, Enomoto [1] has observed that, interchanging the roles of x_1 and y_1 and reversing the orientation of the edge joining them, we obtain another drawing of the graph, in which $(*)$ is again satisfied but not sufficient for the existence of a P_3-partition. This second example is exhibited in Fig. 2.

It is important to note that, though the underlying undirected graphs of the two examples above are isomorphic, they are essentially different instances of the current problem because their vertex partitions are not the same. In notation, we shall write E' for the latter.

In the context of Single Input Double Output controllers, the main question is

Problem 1 *In which classes of graphs and digraphs can the existence of a P_3-partition be decided — and the corersponding search problem be solved — in polynomial time ?*

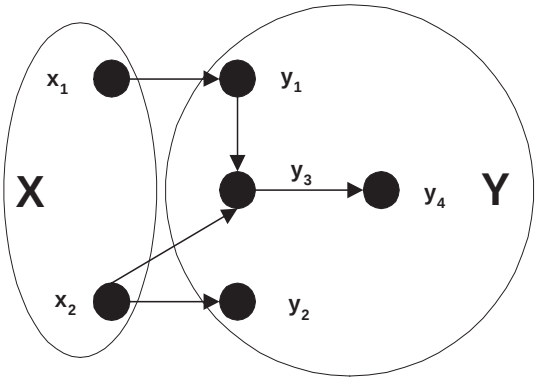

Figure 2: The other vertex partition of E satisfying the Neighborhood-Matching Condition, denoted E'

Later we shall prove that the Neighborhood-Matching Condition can be tested in polynomial time in the class of all (di)graphs; i.e., restricting our attention to (di)graphs satisfying $(*)$ does not help with respect to polynomial-time solvability. On the other hand, the following related question may lead to interesting graph classes.

Problem 2 *What kind of structural properties make the Neighborhood-Matching Condition sufficient?*

A partial answer will be presented in Section 4.

3.2 More general conditions

It is a challenging problem to find further necessary conditions for the existence of a P_3-partition. Instead of restricting our attention to the sets $A \subseteq X$, one may consider any subset $A \subset V$ and require that a collection of vertex-disjoint feasible paths P_3 should exist, which cover all vertices of A. This condition in its generality, however, is already equivalent to the original problem, by choosing $A = V$.

In order to find a more effective approach, one may restrict the selection of the sets A and/or study some parameters weaker than the "P_3-packing number." For the former, a relevant property is as follows:

- We call a set $A \subset V$ *strongly P_3-independent* if no two of its vertices are contained in the same *feasible* P_3 of G.

For instance, the sets involved in the Neighborhood-Matching Condition are of this type, since each feasible path in the P_3-partition is supposed to contain precisely one vertex of X.

For $A \subset V$, let $\mathcal{P}(A)$ denote the set of vertices $v \in V$ for which there exists some $w \in A$ and a feasible P_3 in G containing both v and w. Then $A \subseteq \mathcal{P}(A)$, by the choice $w = v$ for each $v \in A$. Further, we denote by $\tau = \tau(A)$ the minimum cardinality of a vertex set meeting all feasible paths incident to at least one vertex of A.

Proposition 2 *If G is P_3-partitionable, then each of the following conditions is satisfied for all strongly P_3-independent sets $A \subset V$.*

- (i) $|\mathcal{P}(A)| \geq 3\,|A|$.
- (ii) *In particular,* $|A| \leq |V|/3$.
- (iii) $\tau(A) \geq |A|$.
- (iv) *More generally, for every $T \subseteq V \setminus A$, there exist at least $|A| - |T|$ mutually vertex-disjoint feasible paths of length 2 inside $\mathcal{P}(A) \setminus T$.*

•

These simple necessary conditions are strong enough to rule out both E and E' . Indeed, the set $\{y_1, y_2, y_4\}$ is P_3-independent in both graphs (violating (ii)) and the vertex y_3 is contained in all feasible paths, hence yielding an obstruction for any pair of strongly P_3-independent vertices. What is more, the condition (i) is violated by any two of y_1, y_2, y_4, and in E' by the sets $\{x_1, y_2\}$ and $\{x_1, y_4\}$, too. Observe further that if the Neighborhood-Matching Condition does not hold for some $A \subseteq X$, then a suitable set T violating (iv) can be found.

Though we do not know the time complexity of testing (i) in general, we can prove the following related assertion.

Proposition 3 *It can be tested in polynomial time whether the condition (i) of Proposition 2 holds for all subsets $A \subseteq X$.* •

We do not include the proof here because on the subsets of X, (i) is weaker than $(*)$, and we shall show that also the latter can be tested in polynomial time. Hence, in the context of Single Input Double Output controllers, checking (i) inside X is not terribly exciting.

4 On the sufficiency of the Neighborhood-Matching Condition

In this section we first present some classes of graphs where the Neighborhood-Matching Condition is sufficient for the existence of a P_3-partition, and moreover a feasible partition can be found in polynomial time if it exists. Then, in a separate subsection, we show how $(*)$ can be tested in polynomial time in every graph and digraph.

Proposition 4 *If Y is an independent set (i.e., G is bipartite), then the condition $(*)$ is necessary and sufficient for the existence of a P_3-partition. Moreover, if $(*)$ is satisfied, then a P_3-partition can be found in polynomial time.*

Proof. Observe first that if Y induces no edge in G, then $(*)$ reduces to

$$|N(A)| \geq 2\,|A| \qquad \forall\ A \subseteq X\,.$$

Now, double the size of X by taking a copy x' for each $x \in X$ and joining it to the neighbors of x. In this larger bipartite graph, the vertex class containing X satisfies the Hall condition, thus has a perfect matching, say M. As is well known, such an M can be found in polynomial time. Identifying x' with x, M turns to a P_3-partition of G. •

Under the conditions of Proposition 4, $N(X)$ is the entire set Y. We next consider the other extreme, where $N(X)$ is as small as possible.

Proposition 5 *If $|N(X)| = |X|$, then the condition $(*)$ is necessary and sufficient for the existence of a P_3-partition. Moreover, if $(*)$ holds in G, then a P_3-partition can be found in polynomial time.*

Proof. We first show that the search problem is polynomial-time solvable. Note that the assumption above means $|X| = |N(X)| = |Z|$ where $Z := Y \setminus N(X)$. Thus, G admits a P_3-partition if and only if there exist X vertex-disjoint X–Z paths. This holds precisely when both $X \cup N(X)$ and $N(X) \cup Z$ have perfect matchings, and hence the search problem is solvable in polynomial time by standard bipartite matching algorithms.

In order to prove the sufficiency of $(*)$, suppose for a contradiction that one or both $X \cup N(X)$ and $N(X) \cup Z$ do not have a perfect matching. If $X \cup N(X)$ has none, then, by Hall's theorem, there is a subset $A \subseteq X$ with $|N(A)| < |A|$. Since $m(A) \leq n(A)$ holds for every A, we obtain

the contradiction $n(A) + m(A) < 2\,|A|$. On the other hand, if $N(X)$ is not matchable with Z, then we have $m(X) < |X|$, thus the contradiction $n(X) + m(X) < 2\,|X|$ is obtained. •

4.1 Testing the Neighborhood-Matching Condition in polynomial time

The goal of this subsection is to prove that the condition $(*)$ can be checked efficiently. Along these lines, alternative proofs of the previous two propositions can also be obtained; nevertheless, the preceding arguments are definitely simpler than the next one, and, as a more important point, also the search algorithms given there are faster than what could be derived from the construction below.

Theorem 1 *The Neighborhood-Matching Condition can be tested in polynomial time.*

Proof. For each graph or digraph $G = (V, E)$ with $V = X \cup Y$, $|Y| = 2\,|X|$ and X independent, we construct a bipartite graph $G' = (V', E')$ (in linear time) such that the validity of the Neighborhood-Matching Condition in G is equivalent to the existence of a perfect matching in G'. Since the latter can be tested in polynomial time, this construction will prove the theorem.

We define $V' = X_1 \cup X_2 \cup Y_1 \cup Y_2 \cup Y_3$, where the X_i and Y_j are mutually disjoint copies of X and Y, respectively, with their vertices labelled in the same way as in X and Y. The independent vertex classes of G' will be $X_1 \cup X_2 \cup Y_3$ and $Y_1 \cup Y_2$, each of size $4\,|X|$. For all the four combinations of $1 \leq i, j \leq 2$, the edge set joining X_i to Y_j is the copy of the one joining X to Y. Between Y_1 and Y_3 there is a perfect matching, corresponding to the identity mapping. Finally, the neighborhood of a vertex of Y_3 in Y_2 represents the *closed out-neighborhood* of the corresponding vertex in Y; i.e.,

$$y_{3,i}\, y_{2,j} \in E' \quad \Longleftrightarrow \quad i = j \ \vee \ y_i\, y_j \in E\,.$$

We need to prove that the Neighborhood-Matching Condition in G is equivalent to the Hall condition for the vertex class $X_1 \cup X_2 \cup Y_3$ in G'; i.e., to the assumption that each set $Z_1 \cup Z_2 \cup W$ is adjacent to at least $|Z_1| + |Z_2| + |W|$ vertices of $Y_1 \cup Y_2$, where $Z_i \subseteq X_i$ for $i = 1, 2$ and $W \subseteq Y_3$.

We begin with the observation that satisfying the Hall condition in G' for all sets is equivalent to satisfying it for those restricted sets where

- Z_1 and Z_2 are copies of the *same* $Z \subseteq X$, and
- W is a subset of the copy of the neighborhood $N(Z)$ of Z in G.

Indeed, if, say, $x_{1,i} \in X_1$ but $x_{2,i} \notin X_2$, then inserting $x_{2,i}$ into X_2 doesn't increase the neighborhood; i.e., if we had any set violating the Hall condition, we can ensure that there is such a set with the first restriction above, too. Moreover, if W contains some k vertices ($k \geq 1$) which do not belong to the copy of the neighborhood of Z, then removing them from W decreases the neighborhood with k vertices already inside Y_1, hence producing a violating set for the Hall condition with the second restriction as well, if there was any.

Hence, let Z and W be as above. Suppose first that the Neighborhood-Matching Condition *is* valid for the set $Z \subseteq X$. It means that there exists a matching M from $N(Z)$ to $Y \setminus N(Z)$, with $|M| = 2|Z| - |N(Z)|$ edges. We consider the copy of M as represented by the $Y_3 - Y_2$ edges, and now we may even forget about all the other edges incident to W. If W coincides with the entire copy of $N(Z)$, then $Z_1 \cup W$ has $|N(Z)| + |M| = 2|Z|$ neighbors in Y_2, and further $|N(Z)|$ ones in Y_1, that is $|Z_1| + |Z_2| + |W|$ neighbors altogether, as needed. Moreover, the removal of any k vertices from W can destroy at most k neighbors in Y_2 along the copy of M (and no neighbor gets destroyed in Y_1). Thus, the Hall condition holds for all W and Z.

Suppose next that the Neighborhood-Matching Condition is violated by some set $Z \subseteq X$ in G. It means that the size of the largest matching from $N(Z)$ to $Y \setminus N(Z)$ is smaller than $2|Z| - |N(Z)|$. In this situation we are going to apply the following "deficiency version" of Hall's theorem:

> *If the largest matching from a vertex set A to a vertex set B has fewer than m edges, then some subset $W \subseteq A$ has fewer than $|W| + m - |A|$ neighbors in B.*

On applying this to $A = N(Z)$ and $B = Y \setminus N(Z)$ with $m = 2|Z| - |N(Z)|$, and moving then to the graph G', we obtain that some subset W of the copy of $N(Z)$ in Y_3 has fewer than

$$|W| + (2|Z| - |N(Z)|) - |N(Z)| = |W| + 2|Z| - 2|N(Z)|$$

neighbors in Y_2 outside $N(Z_i)$. Thus, together with the neighbors of the $N(Z_i)$ in the Y_i, the set $Z_1 \cup Z_2 \cup W$ has fewer than $2|Z| + |W|$ neighbors, hence violating the Hall condition. This completes the proof of the theorem. •

5 Polynomial algorithms for graphs without E and E'

In this section we prove the following result.

Theorem 2 *There exists a polynomial-time algorithm that decides, for each digraph D containing no subgraphs isomorphic to* E *and* E', *whether D admits a P_3-partition. Moreover, if D is P_3-partitionable, then the algorithm also finds a feasible partition.*

Based on the transformation described at the very beginning of Section 3, such an algorithm can be applied to solve the undirected version of the problem as well (both decision and search), on every input graph G without E and E' as a subgraph. The following standard notation will be used.

Notation and terminology. Oriented edges will be called *arcs*, and the set of arcs in digraph D will be denoted $A(D)$. For two vertex subsets S and T, an (S,T)-*arc* is an arc having its starting vertex in S and its endpoint in T. The *induced subgraph* $D[U]$ has vertex set U, and its edges are all the (U,U)-arcs. Finally, the *out-neighborhood* of vertex v, i.e. the set of vertices w such that vw is an arc, is denoted $N^+(v)$; and we write $d^+(v)$ for the *out-degree* $|N^+(v)|$ of v.

The validity of Theorem 2 will be proved via the following algorithm.

Algorithm A:

Input: A digraph D with the properties described in Section 3, and containing neither E nor E' as a subdigraph.

Step 1: We delete arcs according to the following scheme:

- **1a:** If there are two distinct vertices $x_1, x_2 \in X$ and two distinct vertices $y_1, y_2 \in Y$, such that $N^+(x_1) = N^+(x_2) = \{y_1, y_2\}$, then delete x_1y_2 and x_2y_1.
- **1b:** Otherwise, if there is a vertex $x_1 \in X$, such that $N^+(x_1) = \{y\}$, then delete all arcs into y except x_1y.
- **1c:** Otherwise, if there are two distinct vertices $x_1, x_2 \in X$ and two distinct vertices $y_1, y_2 \in Y$, such that $\{x_1y_1, x_2y_1, y_1y_2\} \subseteq A(D)$, then delete y_1y_2.

1d: Otherwise, if there is a vertex $x \in X$ and two distinct vertices $y_1, y_2 \in Y$, such that $\{xy_1, xy_2, y_1y_2\} \subseteq A(D)$, then delete y_1y_2.

1e: Otherwise, if $x_1y \in A(D)$ is an arc from X to Y, and there is no directed path of length at most 2 from $X \setminus \{x_1\}$ to y, then delete all arcs into y except x_1y.

1f: Otherwise, if $y \in Y$ has no arc from X to y, then delete all arcs out of y.

Note that we check the above steps in the given order. So, for example, if we delete an arc in 1e, then it is because we could not do so in 1a, 1b, 1c, or 1d. Let D' denote the digraph obtained when we cannot delete any further arcs by the above scheme.

This D' satisfies the following structural properties, which we shall prove after the completed description (Step 3) of Algorithm A.

Claim A: The digraph D' has a P_3-partition if and only if D does.

Claim B: There is no simple directed path of length 2 in the induced subgraph $D'[Y]$.

Step 2: If $xy_1 \in A(D')$ is an arc from X to Y, and y_1y_2 is an arc inside Y, then delete y_1y_2 and add the arc xy_2.

Let D'' be the digraph obtained, when we cannot perform the above reduction any more. This D'' will be shown to have the following properties.

Claim C: D'' is bipartite, with independent vertex classes X and Y.

Claim D: D'' has a P_3-partition if and only if D' does.

Step 3: Apply the polynomial-time algorithm described in the proof of Proposition 4, in order to decide if D'' has a P_3-partition — and to find one if it exists — and return the result.

In order to prove that the above algorithm works correctly, we need to prove Claims A, B, C, and D.

Proof of Claim A: Clearly, if D' has a P_3-partition, then the same P_3-partition can be used for D, as D' is a subgraph of D. So assume that D

has a P_3-partition, and let P be the arc set of such a partition, containing the maximum number of arcs in D' (i.e., containing the minimum number of arcs that have been deleted from D in order to obtain D').

First of all, observe that the reductions 1b, 1e, and 1f cannot destroy any directed P_3; i.e., P_3-partitionability is invariant under them. Indeed, in 1b, the only way to cover x_1 is to choose x_1y_1 as the starting arc of a P_3, and then to continue it with an out-going arc from y_1. In 1e, the unique X-vertex that can occur in a P_3 covering y is x, and then the arc xy cannot be continued with any arc oriented towards y. Finally, in 1f, any P_3 covering y must have y as its endpoint (and has to start in X). It follows that P cannot contain any arcs deleted in 1b, 1e, and 1f.

Suppose that xy is a (X, Y)-arc that belongs to P but is not in D'. Then xy must have been deleted in 1a, 1b, or 1e. We have already handled the cases 1b and 1e; hence, assume that xy was deleted in 1a. Without loss of generality, let $xy = x_1y_2$, given the notation in 1a. As some arc starts from x_2 in P, we must have $x_2y_1 \in P$, but now deleting x_1y_2 and x_2y_1 from P and adding x_1y_1 and x_2y_2 to P we obtain a P_3-partition containing more arcs from D' then P, a contradiction.

If yy' is a (Y, Y)-arc that belongs to P but is not in D', then yy' must have been deleted in 1b, 1c, 1d, 1e, or 1f. Again, the cases of 1b, 1e, and 1f have already been settled. Now, if yy' was deleted in 1c, then let $x_1, x_2, y_1 = y$ and $y_2 = y'$ be defined as in 1c, and furthermore we may assume that $x_1y_1 \in P$. As we did not delete x_1y_1 or x_2y_1 in 1b, we must have $d^+(x_1) \geq 2$ and $d^+(x_2) \geq 2$. If $x_1y_2 \in A(D)$, then we may have used x_1y_2, instead of y_1y_2 in P, a contradiction. So there exists $y_3 \in Y - \{y_1, y_2\}$, such that $x_1y_3 \in A(D)$. Now $N^+(x_2) \subseteq \{y_1, y_2, y_3\}$ must hold, since otherwise we obtain E. Therefore, $x_2y_3 \in P$ (as x_2 needs to be covered in P but not matched with y_2), and therefore $x_2y_3 \in A(D)$. Now $N^+(x_1) = \{y_1, y_3\}$, since otherwise we obtain E. As we did not delete any arcs in 1a, we must have $N^+(x_2) = \{y_1, y_2, y_3\}$. Now delete x_1y_1, y_1y_2 and x_2y_3 from P and add x_2y_1, x_2y_2 and x_1y_3 instead, in order to obtain a P_3-partition containing more arcs from D' then P, a contradiction.

If yy' was deleted in 1d, then let $x_1, y_1 = y$ and $y_2 = y'$ be defined as in 1d, and note that x_1y_1 is the only (X, Y)-arc into y_1, as otherwise y_1y_2 would have been deleted in 1c. Therefore $x_1y_1 \in P$, and we may delete y_1y_2 from P and add x_1y_2 instead, a contradiction.

Therefore P is contained in D', and the proof of Claim A is done. •

Proof of Claim B: Let $y_1y_2y_3$ be a simple directed path in $D'[Y]$. There exists a vertex $x_1 \in X$, such that $x_1y_1 \in A(D')$, since otherwise y_1y_2 would have been deleted in 1f at the latest. Analogously, there exists a vertex $x_2 \in X$, such that $x_2y_2 \in A(D')$. Furthermore $x_1 \neq x_2$, as otherwise y_1y_2 would have been deleted not later than in 1d. If $N^+(x_2) \not\subseteq \{y_1, y_2, y_3\}$, then we obtain $\mathsf{E'}$ as a subgraph, a contradiction. However, since we did not delete y_1y_2 or y_2y_3 in 1d, we must have $N^+(x_2) = \{y_2\}$. This implies, however, that y_1y_2 would have been deleted in 1b, a contradiction. •

Proof of Claim C: The set X is independent, by assumption. If y_1y_2 is an arc in D', then there exists a vertex $x \in X$, such that $xy_1 \in A(D')$, since otherwise y_1y_2 would have been deleted in 1f. However, this implies that y_1y_2 is deleted in Step 2 of the algorithm, and therefore Y is independent in D''. •

Proof of Claim D: Let P be the arc set in a P_3-partition of D'. For every (Y,Y)-arc, yy' in P, let $xy \in P$ be the (X,Y)-arc into y in P (which is unique in D', by 1c), and substitute yy' with xy' in P, in order to obtain a P_3-partition of D''.

Now let P be a P_3-partition of D''. We will show that there exists a P_3-partition of D'. If $xy \in P$ does not belong to D', then there exists a vertex $y_1 \in Y$, such that $xy_1, y_1y \in A(D')$. If $xy_1 \in P$, then we can substitute xy with y_1y, so assume that $xy_1 \notin P$. Since Y induces no arc in D'' by Claim C, there exists $y_2 \in Y - \{y_1, y\}$, such that $xy_2 \in P$. Now y_1 must be covered in P and, again by Claim C, the only possibility is an arc $x_1y_1 \in P$, where $x_1 \in X - x$. As $xy_1, y_1y \in A(D')$, we must have $x_1y_1 \notin A(D')$, by 1c. However, since $x_1y_1 \in A(D'')$, there must have been a path $x_1y^*y_1$ in D', where $y^* \in Y$. By Claim B, we must have $y^* = y$. Now we may delete xy and x_1y_1 from P, however, and replace them with xy_1 and x_1y.

Repeating the above transformation as many times as necessary, we eventually obtain a P_3-partition of D'. •

Proof of Theorem 2. The fact that Algorithm A works, follows directly from Claims A and C. Observe further that every step — including the sub-algorithm applied at Step 3 as well — can be done in polynomial time, and no step is done more then a polynomial number of times. Moreover, a solution found for D'' can be transformed to one for D, again in polynomial time. Thus, the running time of Algorithm A is polynomial. •

6 *NP*-completeness results

In this section we prove for two restricted classes of (di)graphs — namely those containing no subgraph isomorphic to E and $\mathsf{E'}$, respectively — that the 1X-2Y problem is *NP*-complete on them. It is worth comparing these results with the theorem of the previous section where we have just proved that the problem is solvable in polynomial time on the intersection of these two graph classes.

Let us note at the beginning that the P_3-partitionability of an arbitrary input graph or digraph clearly is decidable in non-deterministic polynomial time; i.e., 1X-2Y is in *NP*. Hence, in order to prove *NP*-completeness, it will suffice to show that a general instance of some well-known *NP*-hard problem can be reduced in polynomial time to some instance of 1X-2Y which belongs to the particular graph class in question. It will turn out that the directed and undirected graphs can be handled simultaneously; therefore, we shall formulate and prove the two versions of the analogous results together.

As regards terminology, given a fixed *digraph* F, we shall call a digraph G *F-free* if it does not contain any subgraph isomorphic to F. Moreover, an *undirected* graph G will be said to be F-free if none of its subgraphs is isomorphic to the undirected underlying graph of F.

Theorem 3 *The* 1X-2Y *problem is NP-complete on* E*-free graphs, and on* E*-free digraphs, too.*

Proof. We make a reduction from the 3-Dimensional Matching problem (3-DM for short), which is well-known to be *NP*-complete (see, e.g., [2]). An instance of 3-DM is a 3-partite hypergraph $\mathcal{H}$ with a vertex set V partitioned into three mutually disjoint parts V_1, V_2, V_3 of the same cardinality, say $|V_1| = |V_2| = |V_3| = n$, and a collection of 3-element sets $H_1, H_2, \ldots, H_m$ such that

$$|H_i \cap V_j| = 1 \qquad \forall\ 1 \leq i \leq m\,, \quad \forall\ 1 \leq j \leq 3\,.$$

It is *NP*-complete to decide whether the vertex set can be partitioned into n mutually disjoint edges; i.e., whether there exists a partition

$$H_{i_1} \cup \cdots \cup H_{i_n} = V\,.$$

For each instance $\mathcal{H}$ of 3-DM, we construct a (di)graph G in such a way that the vertex set of $\mathcal{H}$ can be partitioned into edges if and only if G

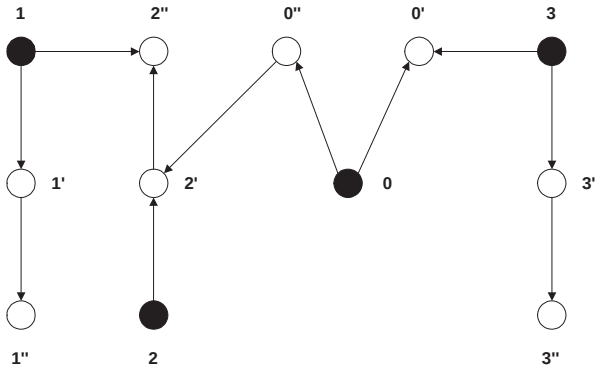

Figure 3: The gadget for reduction from 3-DM; black nodes represent x-vertices

admits a P_3-partition. The construction will be carried out in $O(m+n)$ steps, i.e., linear with respect to input size.

For each $i = 1, \ldots, m$, as exhibited in Fig. 3, consider the digraph F with vertex set

$$V(F_i) = \{x_i^j, y_i^{j'}, y_i^{j''} \mid j = 0, 1, 2, 3\}$$

and edge set

$$\begin{aligned} E(F_i) \;=\; & \{x_i^0 y_i^{0'}, x_i^0 y_i^{0''}, x_i^1 y_i^{1'}, y_i^{1'} y_i^{1''}, x_i^2 y_i^{2'}, y_i^{2'} y_i^{2''}, \\ & x_i^3 y_i^{3'}, y_i^{3'} y_i^{3''}, x_i^1 y_i^{2''}, x_i^3 y_i^{0'}, y_i^{0''} y_i^{2'}\}. \end{aligned}$$

Observe the following structural properties of F_i.

(A) F_i admits a P_3-partition.

(B) If $x_i^0, y_i^{2''}, y_i^{3'}$ are covered with a collection of vertex-disjoint paths P_3, but $y^{3''}$ is not covered, then both x_i^2 and $y_i^{1''}$ remain uncovered, too.

Indeed, (A) is immediately obtained by the four pairs of edges listed in the first two rows in the description of $E(F_i)$; and (B) can be verified taking $y_i^{3'}, x_i^0, y_i^{2''}$ in this order, observing that the path covering x_i^0 must contain $y_i^{2'}$, and therefore the only possibility for covering $y_i^{2''}$ is to take the P_3 with center x_i^1.

Now, for each edge H_i of the input hypergraph $\mathcal{H}$, we make the following identifications:

$$y_i^{1''} = H_i \cap V_1\,, \quad x_i^2 = H_i \cap V_2\,, \quad y_i^{3''} = H_i \cap V_3\,,$$

while making sure that the other nine vertices of F_i remain outside V and, furthermore, the 9-element sets

$$V(F_i) \setminus V$$

remain mutually disjoint for $i = 1, \ldots, m$. Removing the 3-element edges of $\mathcal{H}$ from this structure, we obtain a digraph, which we shall denote by $G = G(\mathcal{H})$. Note that any vertex of G with more than one x-neighbor is non-adjacent to all y-vertices; thus, both G and its underlying graph are E-free.

We claim that G admits a feasible P_3-partition if and only if 3-DM has a solution on $\mathcal{H}$. (As a matter of fact, it is also true that the solutions for $\mathcal{H}$ and for both $G(\mathcal{H})$ and its undirected underlying graph are in one-to-one correspondence.)

Suppose first that $H_{i_1} \cup \cdots \cup H_{i_n} = V$ is a solution of 3-DM on $\mathcal{H}$. Then, as seen above, we can partition each F_{i_ℓ} ($\ell = 1, \ldots, n$) into four paths of length 2, and each of the other $m - n$ subgraphs induced by the sets $V(F_i) \setminus V$ ($n < i \leq m$) into three such paths, hence obtaining a feasible P_3-partition of the entire $G(\mathcal{H})$. Note that any P_3-partition of G feasible in the oriented sense results in a feasible one of the undirected unerlying graph of G as well.

Conversely, suppose that $G(\mathcal{H})$ or its underlying graph is P_3-partitionable, and consider any one of its feasible partitions. Denote by $v_1, \ldots, v_n$ the n vertices of V_3. Note first that these vertices are at distance 2 from the x-vertices of the F_i. For this reason, each of them is covered with a P_3 whose *endpoint* is an x-vertex; moreover, this P_3 must have its two edges inside the same F_i. It follows that each vertex of V_3 is covered in a distinct F_i. We shall assume, without loss of generality, that v_i is covered in F_i, for all $1 \leq i \leq n$.

Hence, there are $m - n$ subgraphs, namely the F_i with $n < i \leq m$, where $y_i^{3''}$ is *not covered* with a P_3 inside F_i. On the other hand, $y_i^{3'}$ is assumed to be covered. Consequently, by the property (B) above, the covering paths inside these F_i do not cover any vertex of $V_1 \cup V_2$. Thus, also the vertices of $V_1 \cup V_2$ are covered with paths inside $F_1 \cup \cdots \cup F_n$ (even though V_2 itself might as well be covered with paths P_3 whose mid-pont is in V_2). Since $|V_1| = |V_2| = n$, and each F_i can cover just one vertex in each of V_1 and V_2, it follows that

$$H_1 \cup \cdots \cup H_n = V\,,$$

i.e., 3-DM has a solution on $\mathcal{H}$. •

Theorem 4 *The* 1X-2Y *problem is NP-complete on* E' *-free graphs, and on* E' *-free digraphs, too.*

Proof. We make a reduction from 3-SAT, one of the fundamental *NP*-complete problems. An instance of 3-SAT is a Boolean formula

$$\Phi = C_1 \wedge \cdots \wedge C_m$$

in conjunctive normal form, over some set of variables, say over the variables $x_1, \ldots, x_n$; and each clause C_j is a disjunction of precisely three literals. (A literal is a variable x_i or its negation $\neg x_i$.)

For each instance Φ of 3-SAT, we construct a (di)graph $G = G(\Phi)$ with the property that Φ is satisfiable if and only if G has a P_3-partition. First, choose an integer $b_i \geq 2$ for each $i = 1, \ldots, n$, such that x_i as well as $\neg x_i$ occur in at most b_i clauses of Φ. Denote

$$b = b_1 + \ldots + b_n \,.$$

The graph G to be constructed will consist of $3m + 2b$ vertices in its set X, $3m + 3b$ vertices in the neighborhood Y_1 of X, and $3m + b$ vertices at distance 2 from X, in the set denoted Y_2. The property of G being E' -free will be ensured by making each pair of vertices in Y_1 non-adjacent.

The first part of X consists of the $3m$ vertices x_i^j for those combinations of (i, j) where a literal of x_i (positive or negative) occurs in clause C_j. We shall sometimes refer to these x_i^j as the "literal vertices" of X. The further $2b$ vertices $x^1, \ldots, x^{2b}$ of X will be called "non-literal." They will play an essential role in creating a P_3-partition from any satisfying truth assignment of Φ.

The neighborhood Y_1 of X consists of three parts:

- a set Y_1' of $2m$ "clause" vertices, two vertices per clause, such that each of the two taken for C_j is adjacent to the three vertices x_i^j belonging to C_j (but is non-adjacent to all the other vertices of X);
- a set Y_1'' of $m + b$ vertices, completely joined to the non-literal part $\{x^1, \ldots, x^{2b}\}$ of X, but non-adjacent to all the x_i^j;
- and a set Y_1''' of $2b$ "cyclic" vertices, whose adjacencies to X will be described later.

Moreover, the $(3m+b)$-element sets Y_2 and $Y_1' \cup Y_1''$ are joined by a *perfect matching* between them.

In order to define the X–Y_1''' adjacencies, it will be conveninent to view Y_1''' as the disjoint union of cyclic sequences

$$(y_1^i, \ldots, y_{2b_i}^i) \qquad \forall\ 1 \leq i \leq n\,,$$

of length $2b_i$ each. By "cyclic" we mean that $y_{2b_i+1}^i$ is considered to be identical to y_1^i. The cyclic sequence indexed with i will belong to the pair $x_i, \neg x_i$ of literals of the same subscript.

Let us call

$$y_\ell^i\, y_{\ell+1}^i$$

an *odd pair* if ℓ is odd, and an *even pair* if ℓ is even. The positive literals x_i will be represented with odd pairs, the negative literals $\neg x_i$ with even pairs. There are b_i (mutually disjoint) odd pairs on cycle i, and the same number b_i of even pairs (which are, again, mutually disjoint among themselves). Hence, by the choice of the b_i, we can assign to each x_i^j a distinct pair, odd or even, depending on whether x_i appears in C_j as a positive or negative literal. We join x_i^j to the two vertices of its assigned pair.

Finally, we define adjacencies between Y_i''' and the non-literal part of X. Since $|Y_i'''| = 2b$, which is precisely the number of non-literal x-vertices, we can make a bijection between those vertices and the (odd and even) pairs $y_\ell^i\, y_{\ell+1}^i$, where i and ℓ run over all feasible values. Let $x(i,\ell)$ denote the vertex assigned under this bijection to the pair $y_\ell^i\, y_{\ell+1}^i$; and make $x(i,\ell)$ adjacent to both y_ℓ^i and $y_{\ell+1}^i$ (but to no other vertex in Y_1''').

Now, Y_1 is an independent vertex set, by definition. Thus, G is E'-free in the undirected sense, therefore all of its orientations will be E'-free as well. To define the *digraph* G properly, we orient all X–Y_1 edges from X to Y_1, and all Y_1–Y_2 edges from Y_1 to Y_2.

It remains to prove that G admits a P_3-partition if and only if Φ is satisfiable.

Suppose first that G has a P_3-partition, and consider any feasible one. The $3m + b$ paths covering Y_2 cover precisely $3m + b$ vertices in each of Y_1 and X. In Y_1, they cover the entire $Y_1' \cup Y_1''$, but none of the cyclic vertices. In the non-literal part of X they cover $m + b$ vertices, leaving $b - m$ uncovered. Finally, among the literal vertices, they cover precisely two of the three x_i^j for each $j = 1, \ldots, m$, leaving just m literals uncovered — one for each clause C_j.

Deleting the vertices having been covered so far, we obtain an induced subgraph on $3b$ vertices (m literals, $b-m$ non-literals, $2b$ cyclic), such that each x-vertex is adjacent to *precisely two* y-vertices, and the latter are mutually non-adjacent. Hence, in the assumed P_3-partition of G, it must be the case that the neighborhoods of those remaining b vertices of X partition Y_1''' into disjoint pairs. Thus, the pairs partitioning the i^{th} cycle are either all odd or all even. If they are odd, we set $x_i = \mathsf{true}$; and if they are even, we set $x_i = \mathsf{false}$. Since each clause contains a literal not covered with the paths covering Y_2, the truth assignment just defined satisfies all clauses of Φ, hence setting Φ to be true.

Conversely, assume that

$$f : \{x_1, \ldots, x_n\} \to \{\mathsf{true}, \mathsf{false}\}$$

is a satisfying truth assignment for Φ. For each clause C_j, we select one x_i^j, whose corresponding literal in C_j sets $C_j = \mathsf{true}$ under f. Note that if some x_i satisfies some C_j, then $\neg x_i$ cannot satisfy any clause; and vice versa. In other words, at most one of the two literals of x_i can be selected over the entire Φ. Now, a P_3-partition of G can be obtained as follows.

- For each of the selected literals x_i^j, we choose its two neighbors in Y_1''' to form a P_3. Since a variable and its negation cannot be selected for Φ at the same time, all the pairs covered this way in any one of the cyclic sequences have the same parity.

- For the other two literals of each C_j we choose two vertex-disjoint paths P_3 with their other endponts in Y_2. These paths cover the entire Y_1', and $2m$ vertices of Y_2.

- In each cyclic sequence, each maximal subsequence of consecutive uncovered vertices has even length. Therefore, the uncovered part of Y_1''' can be partitioned into $b-m$ pairs. Cover each such pair $y_\ell^i y_{\ell+1}^i$ with the P_3 centered at the vertex $x(i,\ell)$. This completes the cover of the cyclic part, and leaves $m+b$ non-literal vertices uncovered.

- The latter $m+b$ vertices can be taken as starting points of $m+b$ paths P_3, which pass through the $m+b$ vertices of Y_2'' and cover the remaining part of Y_2 as well.

In this way, a P_3-partition of G has been derived from the truth assignment f of Φ. •

7 Hard related problems on bipartite graphs

The first problem considered in this section is a frequently occurring variant of the original engineering problem 1X-2Y, while the second one is an innocent-looking twist of Hall's Marriage Problem. Both of them — and also the former when restricted to bipartite graphs — prove to be algorithmically hard.

We formulate the first problem in a similar way as 1X-2Y.

P_3-Covering for Single Input Double Output Controllers (SubX-2Y):

Instance: A graph or digraph $G = (V, E)$, with vertex bipartition $X \cup Y = V$, such that $|Y| \leq 2\,|X|$, $|Y|$ even.

Question: Does there exist a collection of paths P_3, feasible in the sense as in 1X-2Y, yielding a partition of Y?

As one can see, the only difference between the formerly studied 1X-2Y and the present SubX-2Y is that in the latter only a subset of X has to be covered. (In this sense it is similar to SSmE with $m = 2$, but the latter allows to use some edges P_2 as well, beside the paths P_3, for covering Y.) In the original engineering problem it means that there are more input variables available than needed, and we should select a subset of them for stabilizing the system in question.

The following theorem, which is in sharp contrast with the results of Section 5 (since every bipartite graph is both E-free and E'-free), is also a demonstrative example of how hard the concept of packing is, compared to partitioning.

Theorem 5 *The problem* SubX-2Y *is NP-complete already on bipartite graphs, i.e. where every $y \in Y$ has a neighbor in X, and Y is an independent set.*

Proof. We begin with the construction described in the proof of Theorem 4, but make the following little modifications:

- delete the sets Y_1'' and Y_2;
- make Y_1' twice as big, i.e., for each clause C_j take *four* vertices (instead of two), completely adjacent to the three literals x_i^j of C_j.

Now, if some paths of length 2 centered in X generate a partition of Y, the four y-vertices taken for C_j have to be covered with precisely two of the x_i^j. This requires exactly the same as deciding which single x_i^j does *not* take part in the covering of the clause vertices of C_j. Observe that the latter formulation is just what had to be done in the P_3-partitioning of $G(\Phi)$. The only difference is that in the modified new graph, $m + b$ vertices of X remain uncovered (namely those which would be covered with the $m + b$ paths passing through Y_1'' in $G(\Phi)$). Hence, the two problems have the same algorithmic complexity. •

Before formulating the next theorem, let us list some polynomially solvable partitioning problems on bipartite graphs. For this, assume that $G = (V, E)$ is a bipartite graph with vertex bipartition $V = X \cup Y$. The polynomiality of bipartite matching, and its "doubled" version which we have applied also in Proposition 4, mean:

- If $|Y| = |X|$, then it can be decided in polynomial time whether (the vertex set of) G can be partitioned into vertex-disjoint edges.
- If $|Y| = 2\,|X|$, then it can be decided in polynomial time whether (the vertex set of) G can be partitioned into vertex-disjoint paths of length 2.

If $|Y|$ is somewhere "in between," then in a partition of G, some vertices of Y are covered with copies of P_3, and some others just with edges. The next result shows that in this situation the complexity depends very much on whether we are free to choose the part covered with isolated edges, or this part of Y is prescribed.

Theorem 6 *Let $G = (V, E)$ be a bipartite graph with vertex bipartition $V = X \cup Y$, $|X| < |Y| < 2\,|X|$.*

(*i*) *It can be decided in polynomial time whether G admits a vertex partition into paths of lengths 1 and 2, such that every P_3 in the partition has its endpoints in Y. Moreover, if a feasible partition exists, it can be found in polynomial time.*

(*ii*) *If, in addition, a partition $Y = Y' \cup Y''$ is also given, with $|Y'| = 2\,|X| - |Y|$ and $|Y''| = |Y| - |X|$, then it is NP-complete to decide whether there exists a vertex partition of G into $2\,|X| - |Y|$ edges and $|Y| - |X|$ paths of length 2 in such a way that the vertices of Y' belong to the isolated edges.*

Proof.
(i) We are going to prove that this version of the problem can be reduced to finding a largest collection of mutually edge-disjoint paths between two specified vertices of a directed multigraph. The latter is well-known to be solvable in polynomial time.

Given G above, construct a slightly larger graph G^+, on the vertex set $V \cup \{x^*, y^*, z^*\}$. We make G^+ a directed multigraph, as follows.

- The original edges of G remain edges of multiplicity one in G^+ as well, all of them oriented from X to Y.
- There are two parallel edges oriented from x^* to each $x \in X$.
- From each $x \in X$, there is an oriented edge of multiplicity one to z^*.
- From each $y \in Y$, there is an oriented edge of multiplicity one to y^*.
- There are $2\,|X| - |Y|$ parallel edges oriented from z^* to y^*.

We claim that G admits a vertex partition with the properties described in (i) if and only if there exist $2\,|X|$ mutually edge-disjoint paths from x^* to y^* in G^+. (More disjoint x^*–y^* paths cannot exist, because no more edges are incident to x^*.)

If a feasible partition of G exists, then we take the corresponding edges in G^+, moreover the $2\,|X|$ edges incident to x^* and also those to y^*; and, finally, take the edge xz^* for each $x \in X$ covered with just a single edge in the partition of G. It is easily verified that the subgraph constructed this way is the edge-disjoint union of $2\,|X|$ paths of length 3 from x^* to y^*.

Conversely, suppose that there exist $2\,|X|$ edge-disjoint paths from x^* to y^* in G^+, and choose one such set of paths. Denote by E' the set of edges in the union of these paths. This E' contains all edges incident to $\{x^*, y^*\}$, because the edge sets "from x^* to X" and "from $Y \cup \{z^*\}$ to y^*" are edge cuts of size $2\,|X|$. Moreover, the in-degree of any vertex distinct from x^* and y^* is equal to its out-degree; therefore, E' contains precisely $2\,|X| - |Y|$ edges from X to z^*. The endponts of these edges specify a subset $X' \subset X$ of cardinality $2\,|X| - |Y|$. Hence, on applying the equality of in- and out-degrees for all vertices of $X \cup Y$, we obtain that the edges of E' induced by $X \cup Y$ generate a partition on $X \cup Y$, such

that the vertices of X' are covered with single edges, while the vertices of $X \setminus X'$ are covered with copies of P_3.

Consequently, the problem described in (i) can be solved by any polynomial-time algorithm deciding whether there exist $2\,|X|$ edge-disjoint paths from x^* to y^* and finding a feasible collection of them if they exist.

(ii) This part of the theorem follows from the construction given in the proof of Theorem 4. Indeed, it suffices to delete the set Y_2 from the graph $G(\Phi)$. Then, we set $Y' = Y_1' \cup Y_1''$ and $Y'' = Y_1'''$. This smaller graph has a partition feasible under the present conditions if and only if $G(\Phi)$ is P_3-partitionable. Since the latter has already been shown to be *NP*-complete, the same complexity follows for the former, too. •

Finally, we observe that the construction in the proof of Theorem 4 also yields

Corollary 1 *The* 1X-2Y *problem is NP-complete on the instances where both sets Y_i, consisting of the vertices of Y at distance i from X ($i = 1, 2$), are independent and, in addition, the neighborhood of Y_2 contains just $|Y_2|$ vertices.*

8 Multiple Output Controllers

In this short concluding section we show that if the instances are restricted to *bipartite* graphs, then the existence problem of stabilizing structures with Single Input Multiple Output controllers is solvable in polynomial time, for any fixed number m of outputs of the controllers. Let us note that, for polynomial-time solvability, it is essential to allow the controllers to stabilize *fewer than* m states. Indeed, if each of the selected controllers is required to stabilize *precisely* m states, then the problem is *NP*-complete on bipartite graphs for every fixed $m \geq 3$.

Theorem 7 *For every natural number m, the existence problem* SSmE *can be solved in polynomial time on bipartite graphs.*

Proof. We apply a simplified version of the construction described in the proof of Theorem 6(i). Given any bipartite graph $G = (V, E)$ with vertex bipartition $V = X \cup Y$, we adjoin two new vertices x^*, y^*. Orient m newly inserted parallel edges from x^* to each of the $x \in X$, orient the

edges of G from X to Y (each with multiplicity one), and orient one new edge from each $y \in Y$ to y^*.

It is immediately seen that feasible partitions of G into stars with at most m edges each, centered at X, can be extended to collections of $|Y|$ edge-disjoint x^*–y^* paths. Conversely, in any collection of $|Y|$ edge-disjoint x^*–y^* paths, at most m paths pass through each $x \in X$, and each $y \in Y$ is contained in precisely one of the paths. Thus, the feasible partitions of G are in one-to-one correspondence with the path collections of size $|Y|$. If at least one of the latter exists, then one of them can also be found in polynomial time. •

Acknowledgements. Research of the first two authors was supported in part by the Hungarian Scientific Research Fund under grant OTKA T–026575. Part of this work was carried out while the second author visited BRICS, Århus, from where support is gratefully acknowledged, too.

References

[1] H. Enomoto : private communication, September 2000.

[2] M. R. Garey – D. S. Johnson : *Computers and Intractability – A Guide to the Theory of NP-completeness*. Freeman, New York, 1979.

[3] K. M. Hangos – Zs. Tuza : Process Structure Driven Control Structure Selection. In: *Prepr. 13th World Congress of IFAC*, **Vol. M** (1996), pp. 187–192.

[4] K. M. Hangos – Zs. Tuza : Computational Aspects of Graph Theoretic Methods in Control. In: *Computer-Intensive Methods in Control and Signal Processing — Can We Beat the Curse of Dimensionality?* (L. Berec et al., eds.), 2nd European IEEE Workshop, Prague, Czech Republic (1996), pp. 187–192.

[5] K. M. Hangos – Zs. Tuza : Optimal Control Structure Selection for Process Systems. *Computers and Chemical Engineering*, to appear.

[6] R. Holzman : private communication, July 1999.

[7] T. Kailath : *Linear Systems*. Prentice Hall, New Yersey (1980).

[8] A. Kotlov : private communication, July 1999.

Recent BRICS Report Series Publications

RS-01-18 Katalin M. Hangos, Zsolt Tuza, and Anders Yeo. *Some Complexity Problems on Single Input Double Output Controllers*. 2001. 27 pp.

RS-01-17 Claus Brabrand, Anders Møller, Steffan Olesen, and Michael I. Schwartzbach. *Language-Based Caching of Dynamically Generated HTML*. May 2001. 18 pp.

RS-01-16 Olivier Danvy, Morten Rhiger, and Kristoffer H. Rose. *Normalization by Evaluation with Typed Abstract Syntax*. May 2001. 9 pp. To appear in *Journal of Functional Programming*.

RS-01-15 Luigi Santocanale. *A Calculus of Circular Proofs and its Categorical Semantics*. May 2001. 30 pp.

RS-01-14 Ulrich Kohlenbach and Paulo B. Oliva. *Effective Bounds on Strong Unicity in L_1-Approximation*. May 2001.

RS-01-13 Federico Crazzolara and Glynn Winskel. *Events in Security Protocols*. April 2001.

RS-01-12 Torben Amtoft, Charles Consel, Olivier Danvy, and Karoline Malmkjær. *The Abstraction and Instantiation of String-Matching Programs*. April 2001.

RS-01-11 Alexandre David and M. Oliver Möller. *From* HUPPAAL *to* UPPAAL*: A Translation from Hierarchical Timed Automata to Flat Timed Automata*. March 2001. 40 pp.

RS-01-10 Daniel Fridlender and Mia Indrika. *Do we Need Dependent Types?* March 2001. 6 pp. Appears in *Journal of Functional Programming*, 10(4):409–415, 2000. Superseeds BRICS Report RS-98-38.

RS-01-9 Claus Brabrand, Anders Møller, and Michael I. Schwartzbach. *Static Validation of Dynamically Generated HTML*. February 2001. 18 pp.

RS-01-8 Ulrik Frendrup and Jesper Nyholm Jensen. *Checking for Open Bisimilarity in the π-Calculus*. February 2001. 61 pp.

RS-01-7 Gregory Gutin, Khee Meng Koh, Eng Guan Tay, and Anders Yeo. *On the Number of Quasi-Kernels in Digraphs*. January 2001. 11 pp.